Dronă FPV

Introducere

What it is: The project is building a Whoop FPV drone, a small, lightweight quadcopter defined by its compact size and low weight, which gives it extreme maneuverability compared to larger drones.

The starting idea: Initially, I wanted to keep it simple and makeshift, control it with an ESP32 and handle both the controls and camera feed directly from a phone. As I got more into it, though, each proper component started to justify its cost.

How it evolved: For me, FPV immediately brings to mind the goggles, that's the whole point of “first-person view”. So I committed to a proper setup. I also got a dedicated controller, which opened up another advantage: I can now use a flight simulator to practice before the real thing, which should save the drone from a few crashes. Though I got spare parts just in case.

Why it's useful: The use case is niche for now, but that's fine. Personally, I'm hoping it gives me the same feeling of freedom I got the first time I sat behind the wheel of a car and felt like I could go anywhere.

Descriere generală

TODO:

- Software block diagram
- Describe both diagrams

Hardware Design

Drone Parts:

- 1x Frame
- 1x Canopy
- 4x Motors
- 1x AIO (FC + ESC + RX + VTX)
- 1x Analog Camera

- 1x Antenna
- 1x 1S LiHV Battery (+ 3x reserves)
- 2x CW Propellers
- 2x CCW Propellers

Misc. Parts:

- 1x Lipo Safe Bag
- 1x Analog FPV Goggles
- 1x Radio Control Transmitter

Software Design

Description of the firmware

Custom flight controller firmware for the Happymodel X14 AIO board (STM32G473CEU6, Cortex-M4F @ 170 MHz). The firmware reads stick commands from a Radiomaster Pocket transmitter over an ExpressLRS 2.4 GHz link (CRSF protocol), reads angular rates from the on-board ICM-42688-P IMU, runs a rate-mode PID controller, mixes the result onto four motors via the DShot300 ESC protocol, and in parallel drives the analog OSD overlay (AT7456E), the video transmitter (SmartAudio), the battery ADC, and arming/failsafe safety logic.

Pure C11, no RTOS. A single super-loop with a cooperative scheduler dispatches tasks at fixed rates (IMU 8 kHz, control 4 kHz, RX 1 kHz, OSD 30 Hz, etc.); all I/O is interrupt or DMA-driven.

Development environment

- **IDE:** JetBrains CLion 2026.1.2
- **Build system:** PlatformIO Core
- **Framework:** stm32cube (STM32CubeG4 HAL + CMSIS)
- **Board definition:** a custom boards/g473_custom.json written based on the datasheet
- **Flashing:** STM32G4 USB ROM bootloader (DFU), no external programmer needed

Libraries and third-party sources

- **STM32Cube HAL (STM32CubeG4)** STMicroelectronics' official low-level library
- **CMSIS-Core** ARM's standard Cortex-M abstraction
- **STM32G4 ROM DFU bootloader** not a library, but a third-party-supplied flashing path used in place of writing a custom bootloader
- **dfu-util** open-source command-line tool invoked by PlatformIO to push firmware to the ROM bootloader

- **Protocol specifications** (data only, no imported code): CRSF (ExpressLRS/TBS), DShot300 (Betaflight wiki), SmartAudio v2, AT7456E datasheet, ICM-42688-P datasheet

Data structures

- **Blackboard** A single global state_t struct holds every cross-module value: RC channels, IMU samples, PID setpoints, motor outputs, battery, arm/failsafe flags, and diagnostic counters (worst-case RC gap, failsafe rising-edge count, last disarm cause).
- **Task table** Fixed-size array of tasks (functions) indexed by a task_id_t enum. Each iteration walks the table, calling any task whose next_due_us has passed and advancing it.
- **CRSF parse buffer** Fixed 64-byte frame buffer fed one byte at a time from the USART3 RX interrupt.
- **CRSF state machine** WAIT_SYNC → READ_LEN → READ_PAYLOAD → VALIDATE
- **DShot DMA buffer** Per-motor 20-word array (16 frame bits + 4 reset-gap entries) written into TIM2 CCRx via DMA1 channels 1–4.
- **OSD frame buffer** Small array of (row, col, str) overlay items rebuilt each frame and pushed into the AT7456E character RAM over SPI2.
- **Per-axis PID state** Static array of three {integ, prev_err, gains} records. No dynamic allocation anywhere in the firmware.

Algorithms

- **Cooperative scheduler** Monotonic 32-bit microsecond clock from the DWT cycle counter; uses unsigned subtraction so the comparison stays correct across the ~71 minute wrap.
- **PID rate controller** Per-axis discrete PID with integrator clamp and output clamp. Integrator and derivative state are reset on every disarm → arm transition to avoid wind-up from pre-arm stick movement.
- **Gyro bias auto-calibration** 1000-sample running average of the gyro signal at boot while the drone is stationary; subtracted from every subsequent sample. Calibration is gated by an IMU-still check before the FC is allowed to arm.
- **Quadcopter mixer** Standard X-configuration mix of throttle/roll/pitch/yaw onto four motors. Outputs are scaled by throttle so the controller cannot demand torque at idle; if any motor saturates, the throttle floor is raised to preserve control authority.
- **DShot300 bit encoding** 11-bit throttle + 1-bit telemetry-request + 4-bit XOR CRC, encoded as PWM duty 375/1000 for a “1” and 188/1000 for a “0”, emitted bit-perfect by TIM2 + DMA with no CPU involvement.
- **CRSF parser** Byte-fed state machine with CRC-8 (poly 0xD5) validation, 11-bits-per-channel unpacking, and link-statistics decode (RSSI, LQ, SNR).
- **Failsafes** Various safety checks (RC timeout > 250 ms, low battery, etc.). If any trigger, motors are commanded to zero and the FC is force-disarmed.
- **Arming gate** Bitmask `arming_block` aggregates the set of reasons arming is currently denied (boot lock, IMU not calibrated, low battery, failsafe active, arm switch already up at boot).

Sources

- **GIT**: https://github.com/Coman-Madalin/FPV-Drone-Firmware

Rezultate Obţinute

TODO: Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

29 April 2026

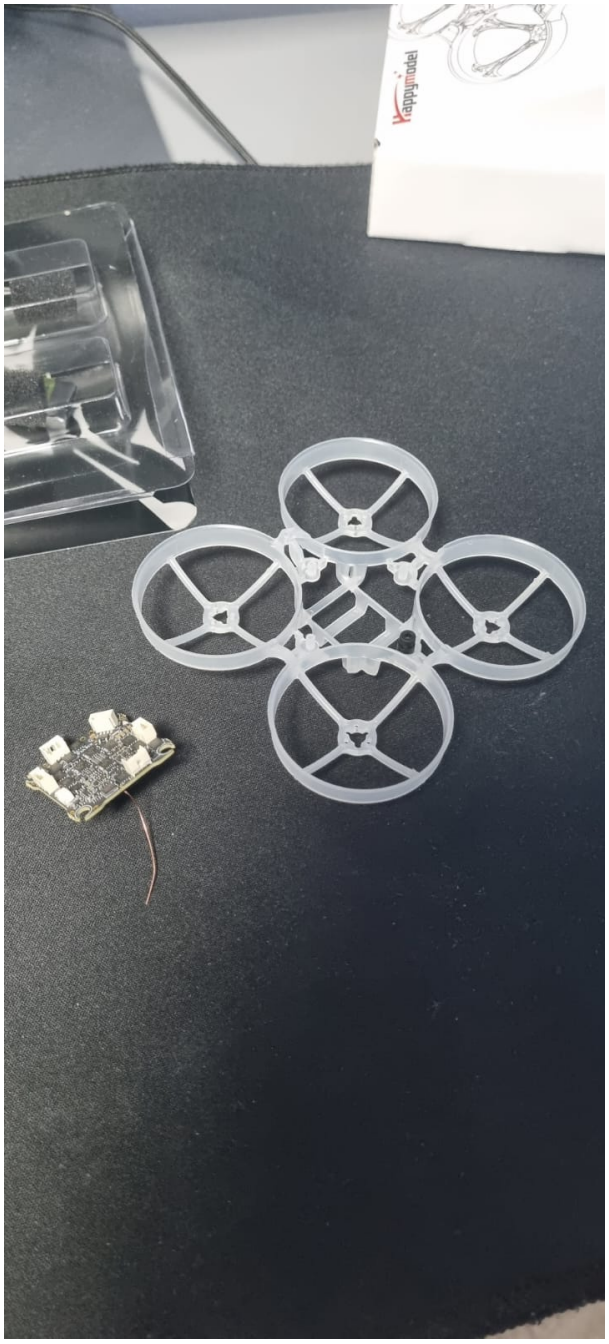
- Placed parts order

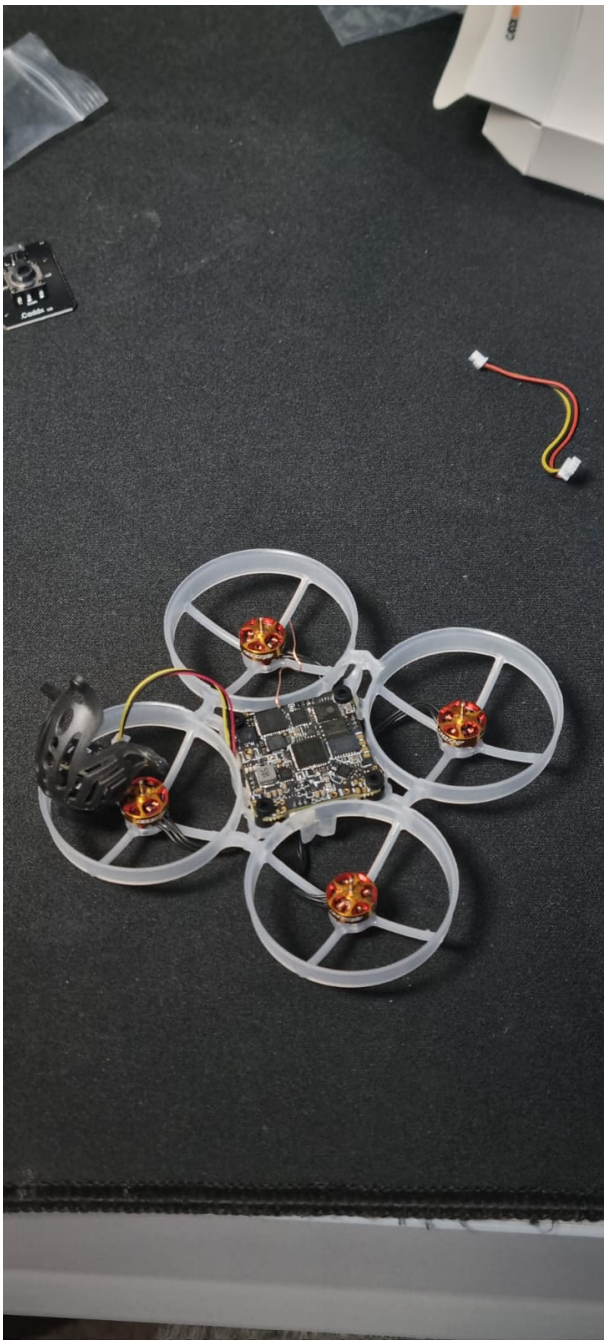
14 May 2026

- Received parts order

16 May 2026

- Finished hardware side


Happymodel



23 May 2026

- Finished Firmware

Bibliografie/Resurse

https://www.happymodel.cn/index.php/2025/10/09/x14-elrs-5-in-1-aio-flight-controller-built-in-2-4g-uart-elrs-v3-0-and-openvtx/

https://www.st.com/resource/en/datasheet/stm32g473cb.pdf

https://deepwiki.com/ExpressLRS/ExpressLRS/3.1-crsf-protocol-and-router

https://betaflight.com/docs/wiki/guides/current/Dshot

https://www.team-blacksheep.com/media/files/tbs_smartaudio_rev09.pdf

https://product.tdk.com/system/files/dam/doc/product/sensor/mortion-inertial/imu/data_sheet/ds-000347-icm-42688-p-v1.6.pdf

Export to PDF